Software Modeling Development with UML (CSP 586)

Project Report

Use generative AI to recommend social places (restaurants bars, etc.) based on the user profile, locations, posts, search histories.

Team Information

Group Number - 21

Batra, Karan (A20518491)
Bisurkar, Shivdeep (A20525712)
Kamble, Soham (A20517098)

Table of Content

1. Project Overview

In an age where technology pervades every part of our lives, the need for individualized experiences has become critical. The project aims to change how people find and interact with social locations like restaurants, bars, and cafés by developing an innovative AI-powered recommendation engine.

At its heart, the initiative aims to apply powerful AI algorithms to personalize suggestions to each user's individual interests and surroundings. The system seeks to understand the intricacies of individual interests and needs by diving into the complex web of user profiles, locations, postings, and search history. It aims to move beyond the one-size-fits-all approach of standard recommendation systems, instead providing a personalized experience that connects profoundly with each user.

This attempt offers more than simply technological innovation; it symbolizes a vision of empowerment and enrichment for users who want to traverse the social world confidently and easily. The project aims to allow users to find new and intriguing social venues that match their tastes and interests by harnessing the immense reservoir of data available to us and the unique powers of AI.

The project intends to push the boundaries of traditional recommendation systems by seamlessly integrating cutting-edge AI models, complex data analytics, and intuitive user interfaces. It aims to build a dynamic ecosystem that combines accidental discoveries with customized experiences, encouraging a sense of connection and belonging in an increasingly digital world.

In other words, by developing an advanced AI-powered recommendation engine, this initiative intends to transform how consumers discover social spaces like restaurants, bars, and cafés. Using cutting-edge generative AI techniques, the system will scan user profiles, locations, postings, and search history to provide personalized suggestions based on individual interests and context.

2. Project Requirements

1. Integration of OpenAI Model: OpenAI's sophisticated model will fuel the recommendation engine, allowing for complete analysis of user data to create relevant and tailored suggestions. By analyzing user profiles, locations, postings, and search histories, the system will give suggestions that are tailored to each user's specific tastes and interests.

2. Utilization of Yelp API and Google Maps API: The project will use Yelp and Google Maps' vast databases via their respective APIs to acquire complete information on social areas. These APIs, which incorporate user preferences, will make it easier to construct a curated list of appropriate suggestions based on characteristics such as geography, cuisine, atmosphere, and user reviews.

3. User Profile Management: Users may establish and maintain their profiles, including preferences for cuisines, dietary restrictions, and ambiance. Allow users to change their profiles and preferences as needed.

4. Data Integration: Use publicly accessible datasets, such as the Yelp API, to enhance the platform's database with information on restaurants, bars, and events, such as locations, reviews, ratings, and other pertinent data. Ensure that data privacy standards are followed and put in place safeguards to secure user data and prevent unwanted access.

5. Search Functionality: Implement a basic search function that allows users to find services, places, events, goods, and more, with filtering options based on user preferences such as cuisine type, price range, distance, and reviews.

6. Geographic Search: Enable geographic search capabilities, which allows users to look for services, places, and events within a certain zip code or radius. Integrate maps and geolocation services to graphically show search results and allow users to filter them based on location and other criteria.

7. User Interaction: Allow users to engage with search results by seeing information, reading reviews, checking ratings, and leaving reviews, ratings, and comments about locations they've visited. Implement ways for users to submit feedback on the platform and its recommendations.

8. Accessibility and Usability: Make sure the platform is user-friendly and accessible across all devices and screen sizes, including features that allow users with impairments to browse and engage with the program. Make the user interface intuitive and simple to browse.

3. Project Features

1. Implementation of the 'Plan My Day' Feature: In addition to customized suggestions, the system will have a distinct 'Plan My Day' function. This feature will create customized itineraries for users, taking into account their interests, time limits, and geographical location, in order to improve the user experience and social exploration.

2. Analytics Dashboard using Google Charts API: To give insights into user behavior, preferences, and system performance, an analytics dashboard will be built with the Google Charts API. This dashboard will provide visuals and analytics to help with system optimization and decision-making, assuring ongoing development and refinement.

3. Refined Searches, Better Choices (Get Suggestions on Keywords): Customize your search by price, location, keyword, and category to find the ideal events and venues, from eateries to concerts, tailored just for you.

4. Smart Chatbot for Users: Engage with our smart chatbot that makes finding your next hangout spot as easy as having a conversation. It's built to understand your likes and offer suggestions that feel personally curated for you.

5. Integration of a Voice Assistant with Whisper (OpenAI Library): Finally, the project will construct a voice assistant powered by Whisper, an OpenAI Library. This voice assistant will provide hands-free interaction with the recommendation system, allowing users to inquire for suggestions, modify preferences, and access system functions via natural language commands.

6. User-Friendly Interface Design: An important part of the project is to design and construct an intuitive and visually appealing user interface. This interface will allow users to easily input their preferences, browse suggestions, and maintain their profiles.

4. Use Cases

Use cases are unique scenarios or settings that describe how users interact with a system or application to accomplish specified objectives or activities. They provide a framework for understanding customer needs and directing the design and development process. Use cases are important because they give a clear knowledge of user demands and assist to guarantee that the final product fits those needs effectively and efficiently. Below is the list of use cases based on different personas in our project:

Users:

- Perform keyword searches for restaurants, bars, events, etc.
- Filter search results based on preferences like cuisine type, price range, and rating.
- Explore top-rated places and events through visualizations on the dashboard.
- Interact with the dashboard to delve deeper into search results.
- Use geospatial search to find nearby places within a specific zip code or radius.
- View search results on a map and filter them based on location.
- Explore review counts and ratings per zip code through visualizations on the dashboard.
- Create and manage their profiles, specifying preferences such as cuisine type and dietary restrictions.
- Provide feedback on recommendations and leave reviews, ratings, and comments for visited places.

Business Owners:

- Manage their business profiles on the platform, including updating information and responding to user reviews and comments.
- Access analytics and insights on their business performance.
- Collaborate with administrators to resolve any issues related to their listings.

Guest:

- Browse through the platform's content without registering or logging in.
- Perform simple searches for places, events, etc.
- View basic information about businesses and events.
- Access public feedback and reviews without the ability to leave comments or ratings.
- View search results on a map and filter them based on location.

Moderator:

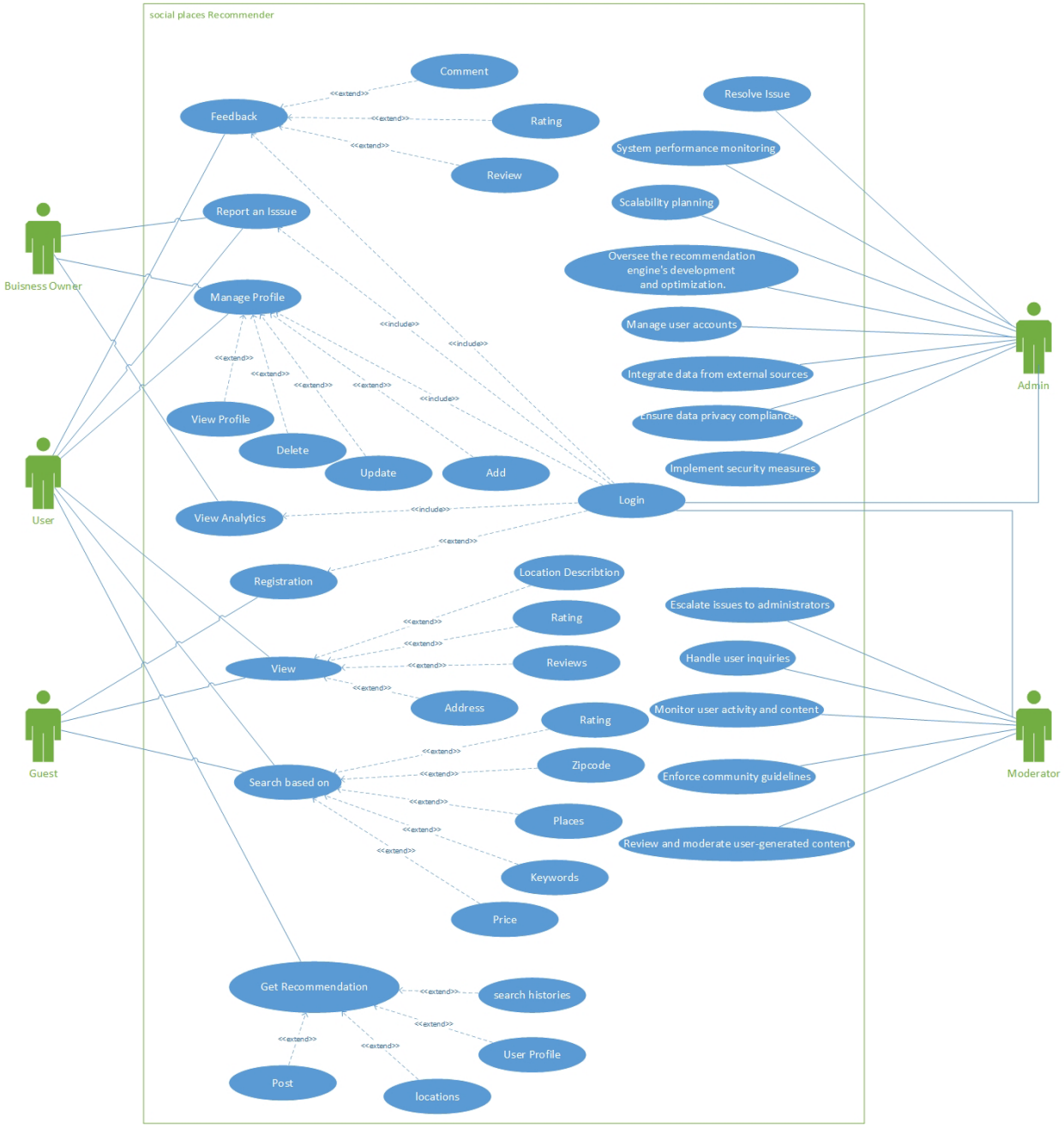
- Monitor user activity and content on the platform.
- Review and moderate user-generated content such as reviews, ratings, and comments.
- Ensure compliance with platform guidelines and policies.
- Handle user inquiries and escalate issues to administrators if necessary.

Admin:

- Manage user accounts, including registration, authentication, and permissions.

- Integrate data from external sources such as Yelp API.
- Ensure data privacy compliance.
- Implement security measures to protect user data and prevent unauthorized access.
- Oversee the recommendation engine's development and optimization.
- Handle system performance monitoring and scalability planning.

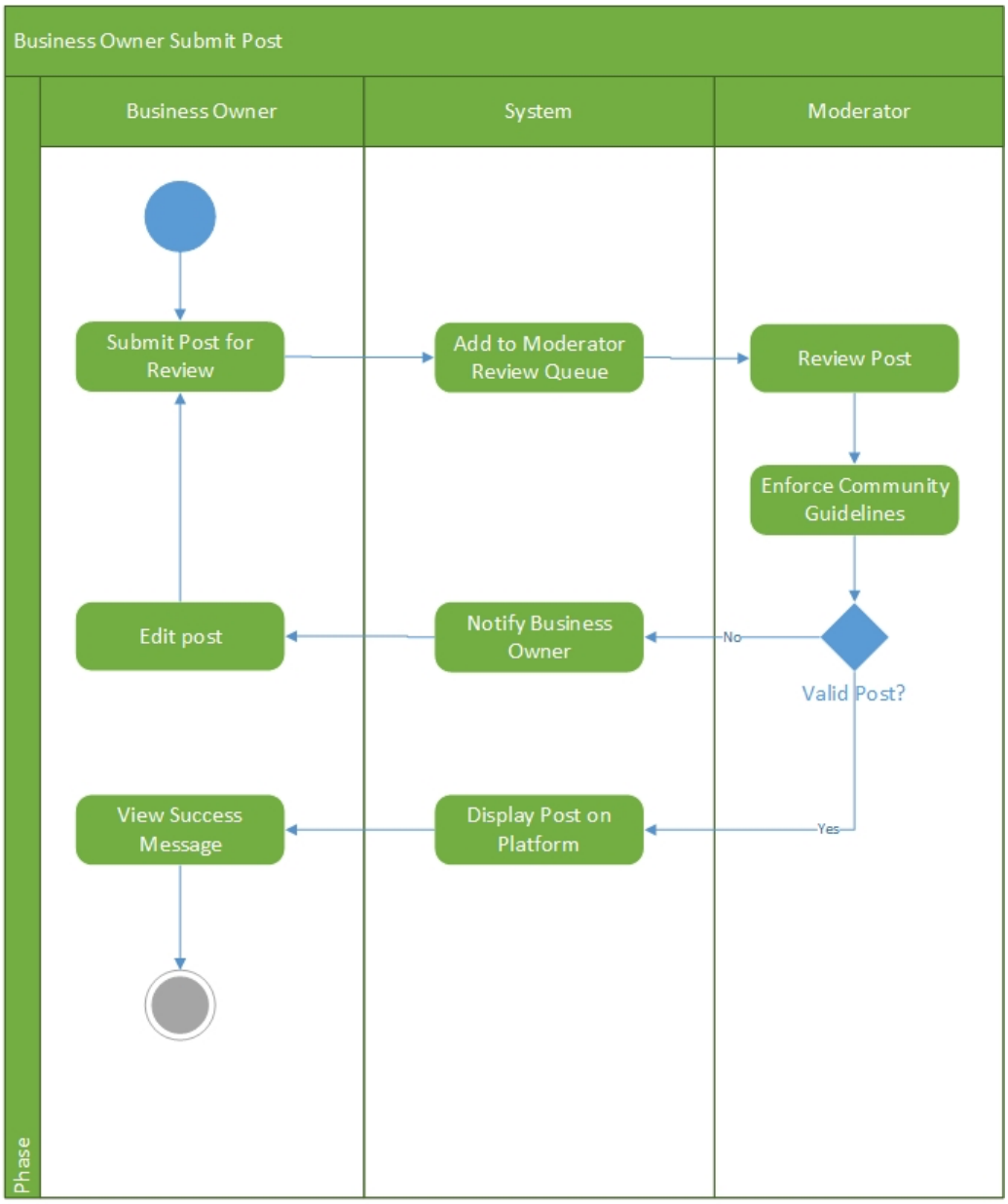
4.1 Use Case Diagram



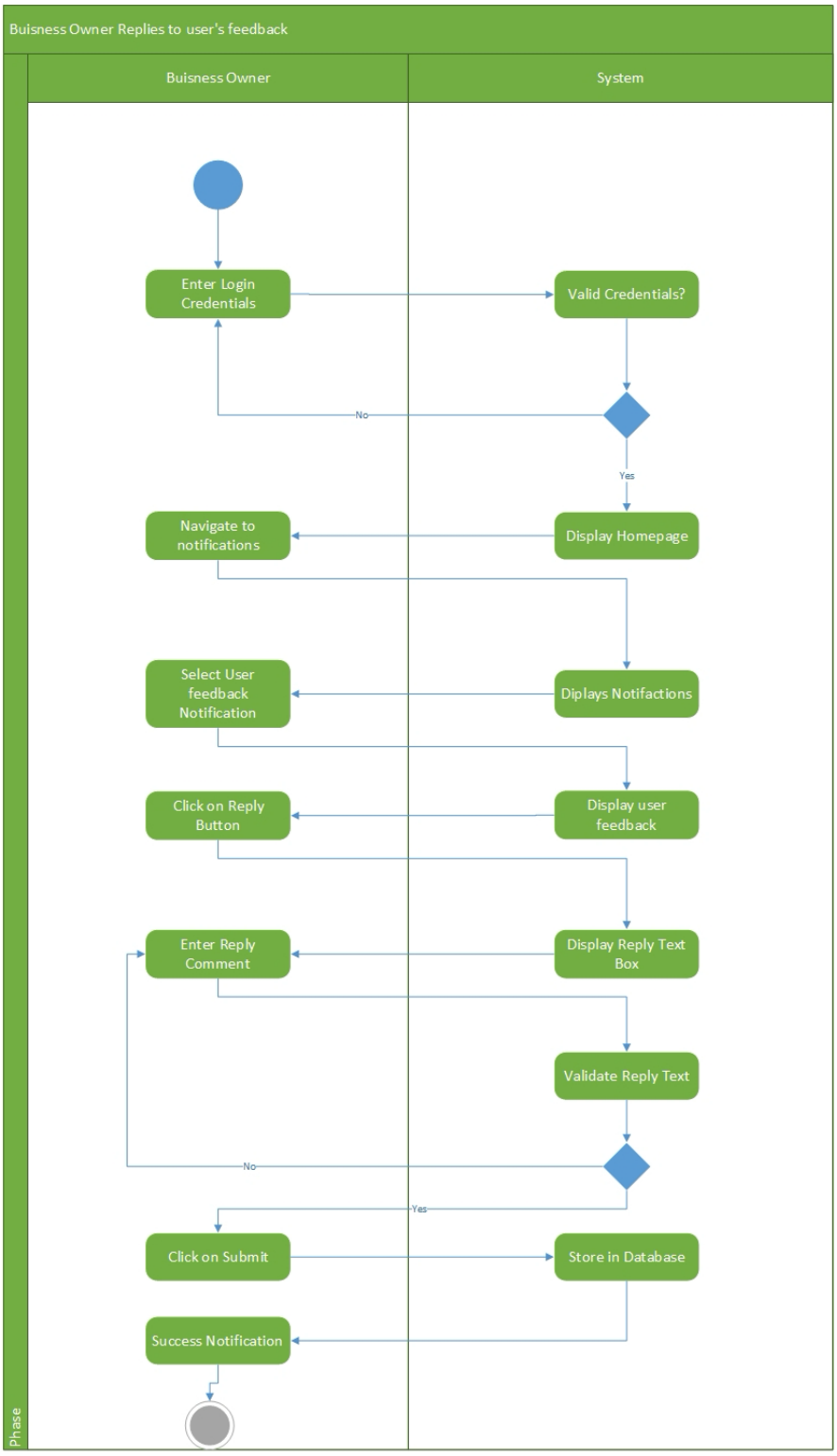
5. Activity Diagrams

Activity diagrams are graphical representations that show the flow of activities inside a system, process, or workflow. They give a visual representation of the sequence of events or processes involved in completing a certain activity or objective. Activity diagrams aid in understanding a system's dynamic elements, such as the sequence of execution, decision points, branching routes, and concurrent tasks.

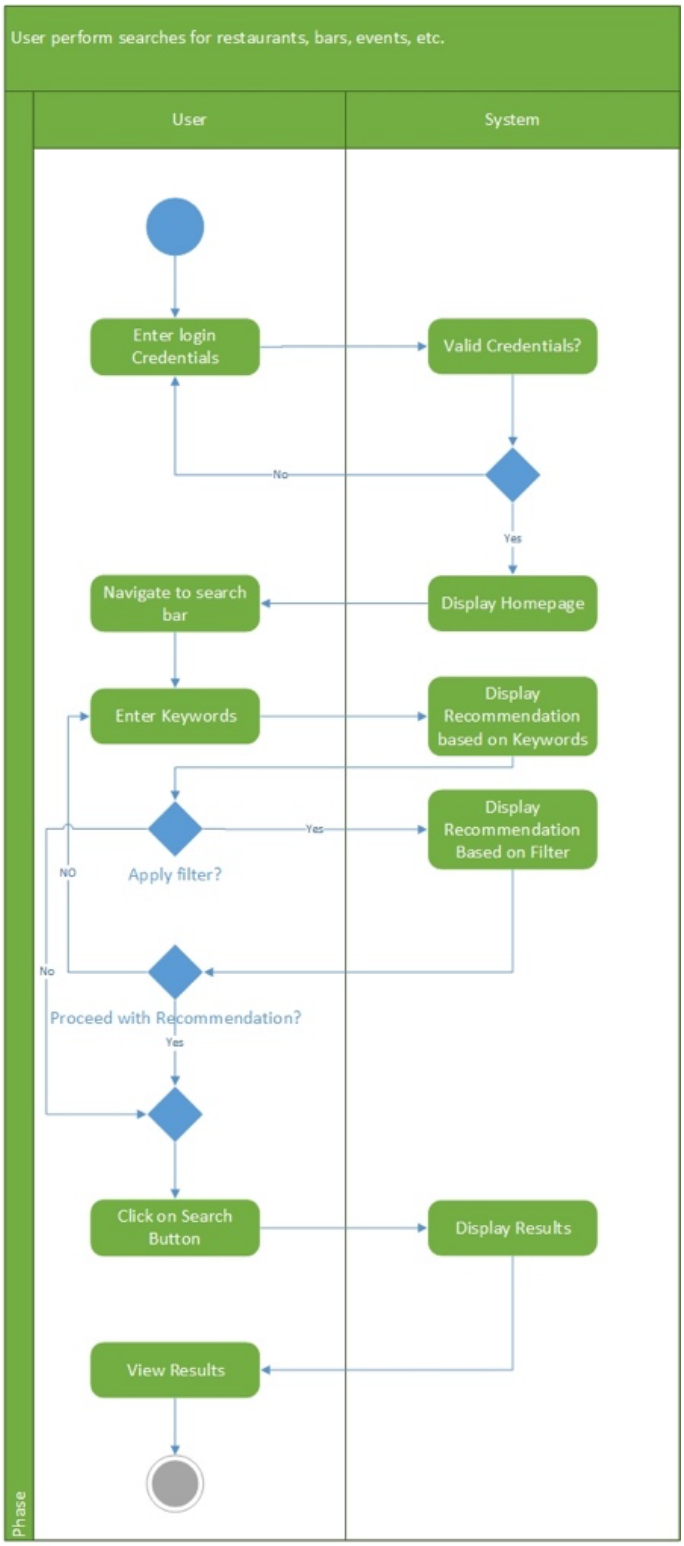
1. Business Owner Creating a New Post: This diagram outlines the moderation workflow where a business owner submits a post, a moderator reviews it against community guidelines, and it's either posted or returned for edits.



2. Business Owner Replying to User's Feedback: The activity diagram maps out the process a business owner follows to respond to user feedback within a system, from logging in to submitting their reply which is then validated and stored in the database.



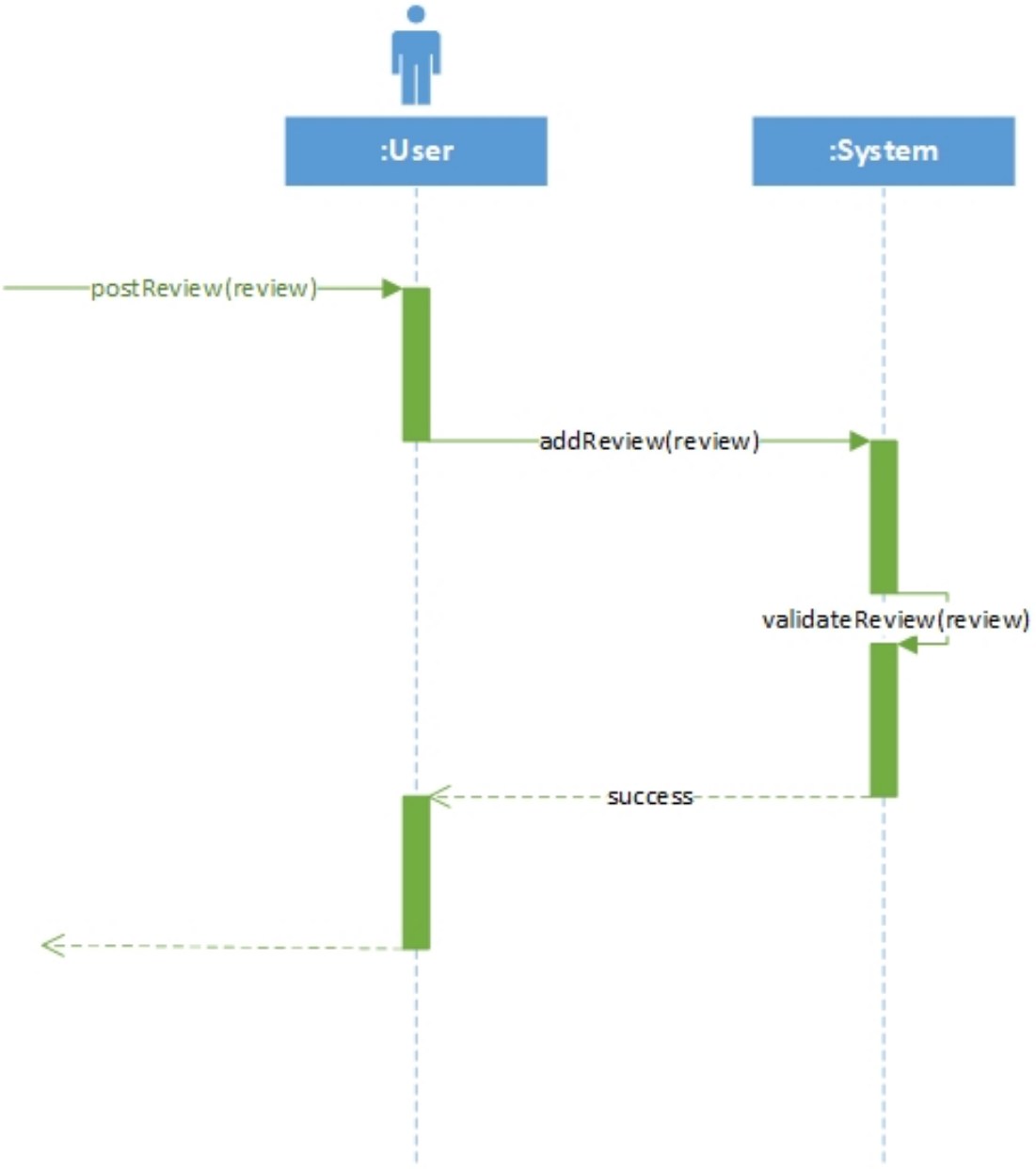
3. User Performing Searches for Places: The activity diagram provides a visual representation of the user's search process flow within a digital platform, from login to viewing the filtered results for restaurants, bars, and events.



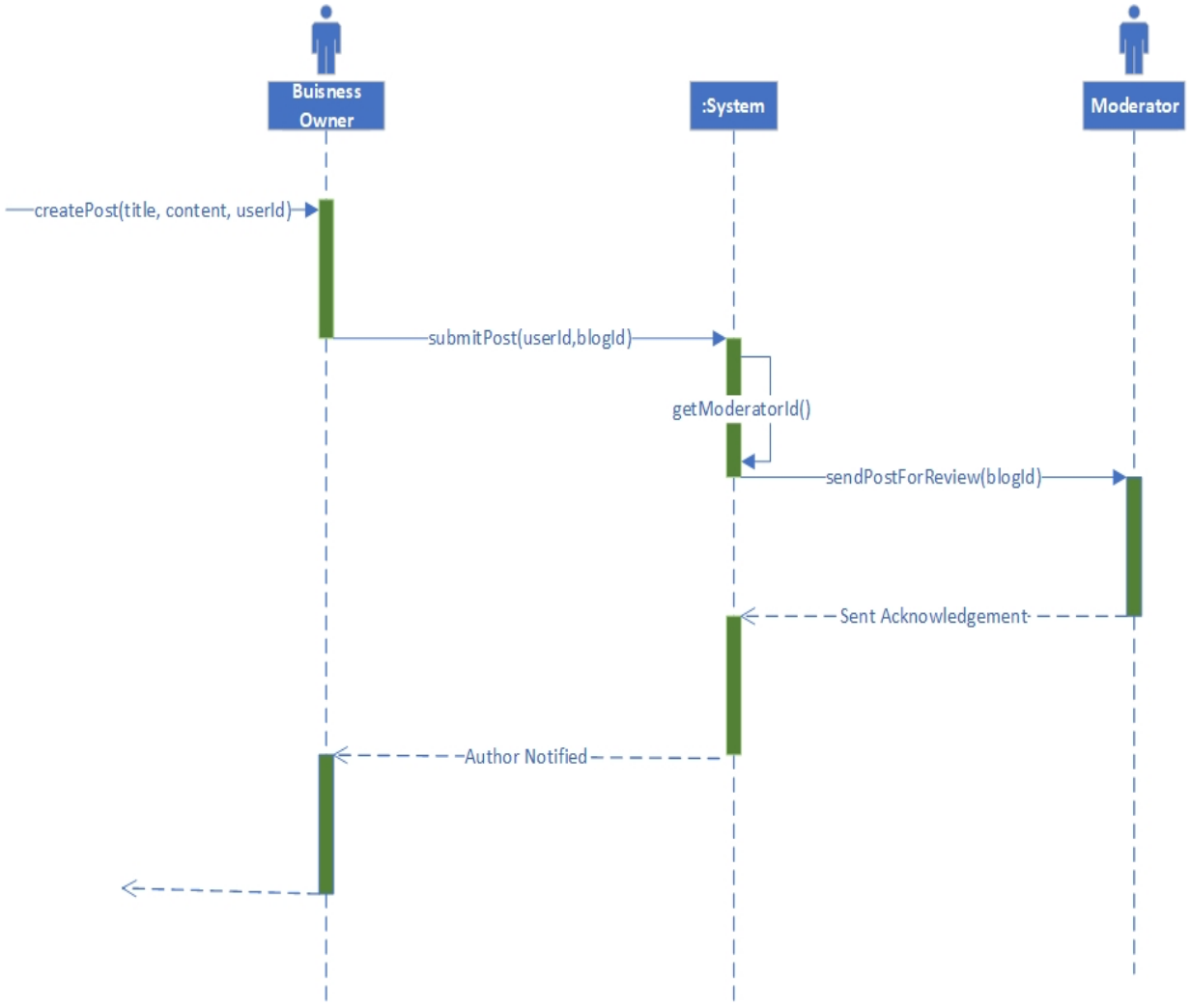
6. Sequence Diagrams

Sequence diagrams are visual representations of the interactions between items or components in a system across time. They represent the chain of messages exchanged between objects in chronological order, demonstrating how they collaborate to complete a certain job or circumstance. Sequence diagrams aid in understanding a system's dynamic behavior, such as control flow and the timing of object interactions.

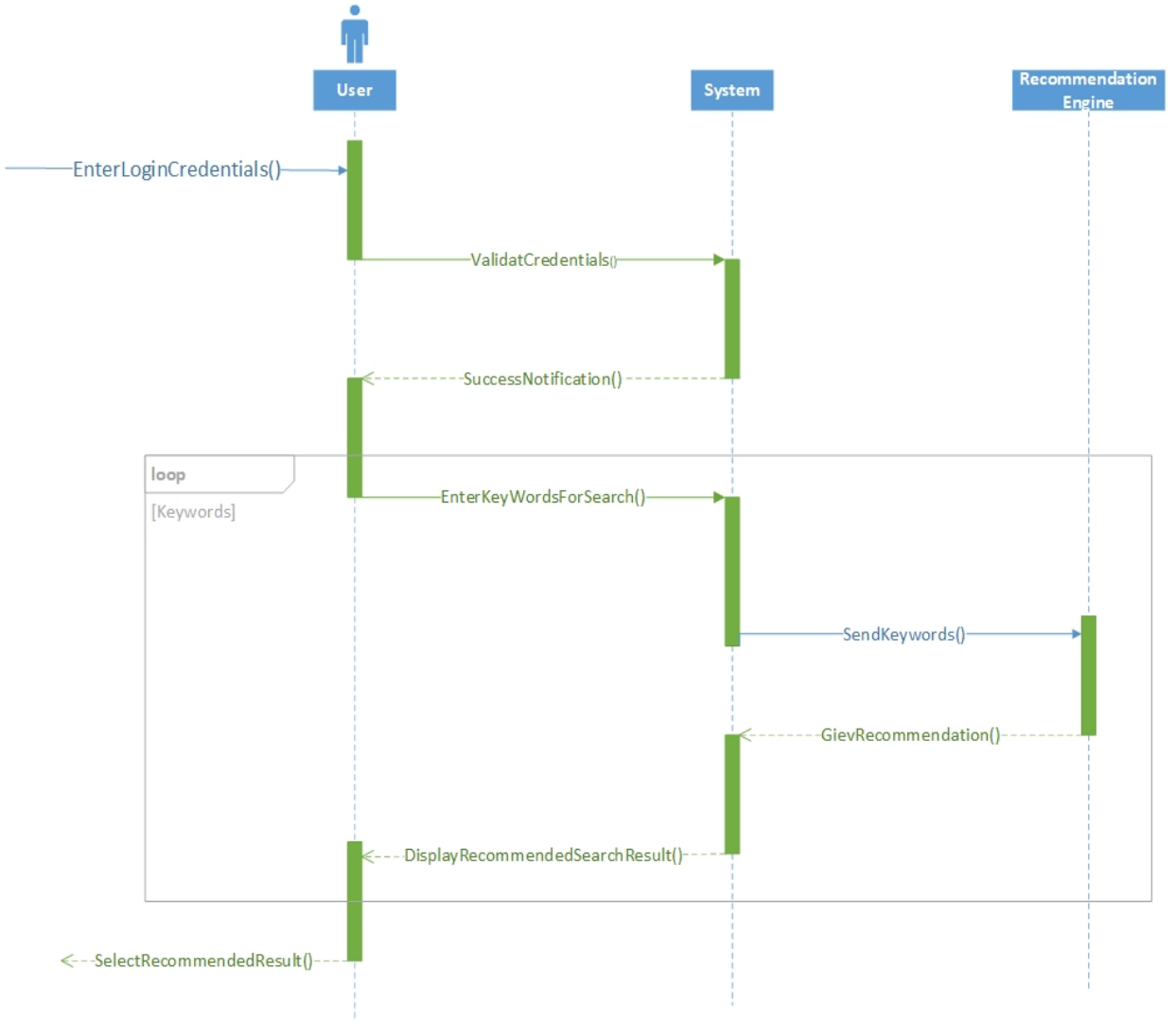
1. Post a Review: This sequence diagram illustrates the process where a user posts a review, and the system adds and validates the review before acknowledging the action's success.



2. Moderator Reviewing Posts: The sequence diagram shows a business owner creating a post, the system processing and submitting it to a moderator for review, and then sending an acknowledgment back to the author.



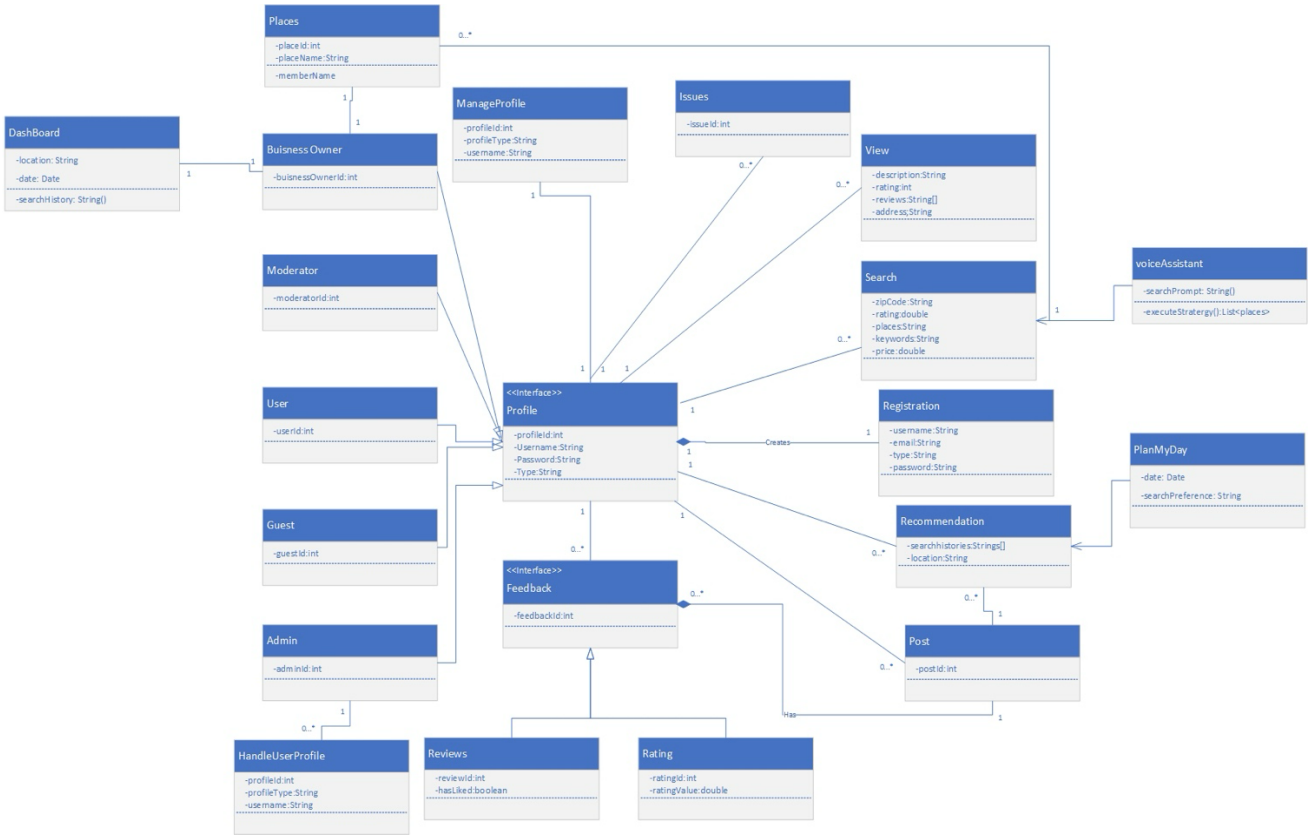
3. User Generating Recommendations: This sequence diagram details the interactions between the user, system, and recommendation engine during the login process and keyword-based search leading to tailored search results.


User
System
Recommendation Engine
EnterLoginCredentials()
ValidatCredentials()
SuccessNotification()
loop
[Keywords]
EnterKeyWordsForSearch()
SendKeywords()
GievRecommendation()
DisplayRecommendedSearchResult()
SelectRecommendedResult()

7. Domain Model Class Diagram

A domain model is a conceptual representation of the concepts, entities, and interactions found in a certain issue domain or topic area. It emphasizes on capturing the key concepts and their relationships, regardless of precise implementation specifics or technological issues. Domain models may employ class-like structures to describe domain entities and characteristics, but they are not standard UML class diagrams.

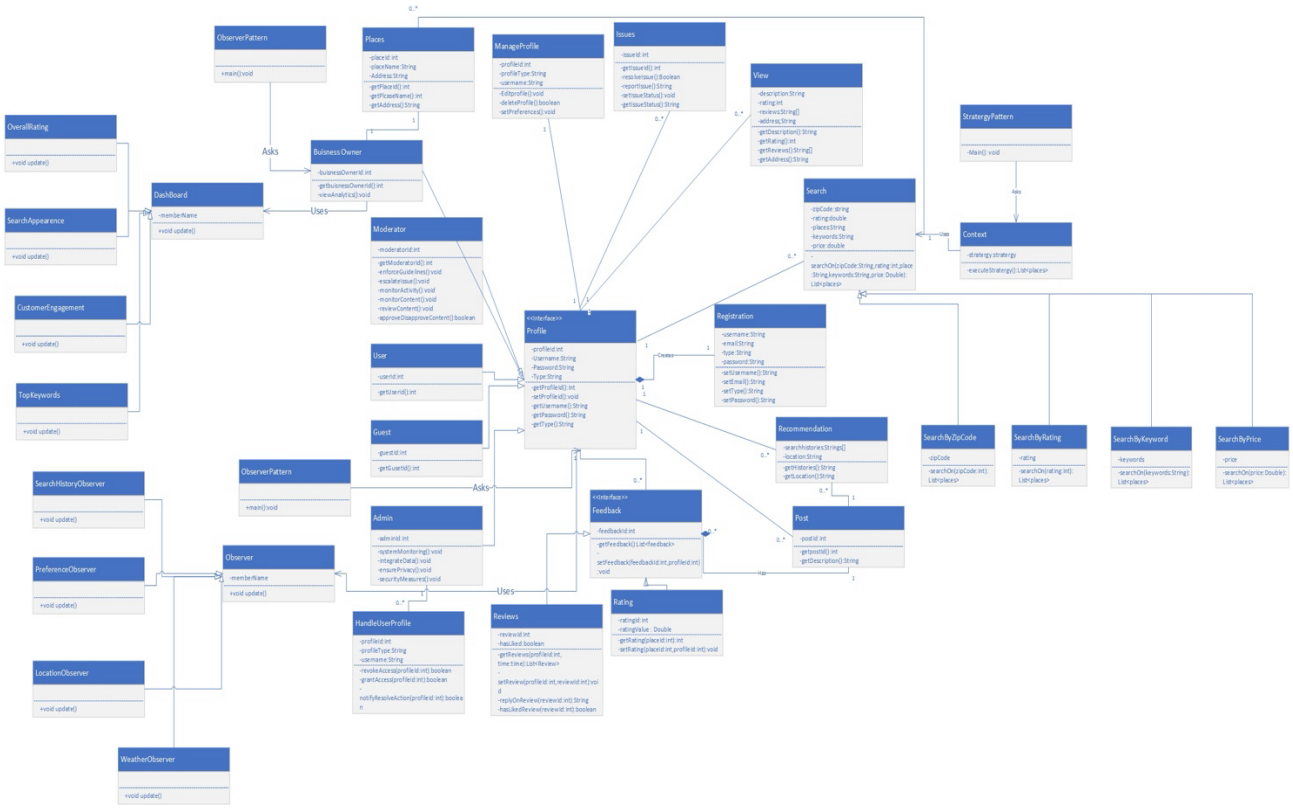
The domain class diagram captures the conceptual framework of the platform, focusing on the roles and responsibilities of entities such as Business Owner, Moderator, and User, and their interactions with system functionalities like Search and View. This diagram lays the groundwork for the detailed design phase, defining the system's structure with elements like Profiles, Issues, and Recommendations, and showing the relationships and data flow without specifying technical implementations. It serves as the blueprint for understanding how different actors engage with the platform's core features, from registration to posting and feedback mechanisms.



8. Design Model Class Diagram

In contrast, a design model represents a software system's full design and implementation, including class structures, interfaces, connections, and behavior. Class diagrams are frequently used in design models to describe the system's static structure, including classes, properties, methods, and connections. These class diagrams follow UML standards and conventions and are used to explain design decisions and system architecture to stakeholders and developers.

The design class diagram represents a location-based search platform, detailing interactions among users like Business Owner and Moderator, and employing the Strategy Pattern for diverse search methods. It integrates the Observer Pattern to update various metrics like Overall Rating and Search Appearance, illustrating a reactive system that adapts to user activities and preferences. This object-oriented structure emphasizes modularity and encapsulates user profiles and feedback mechanisms for a robust user experience.



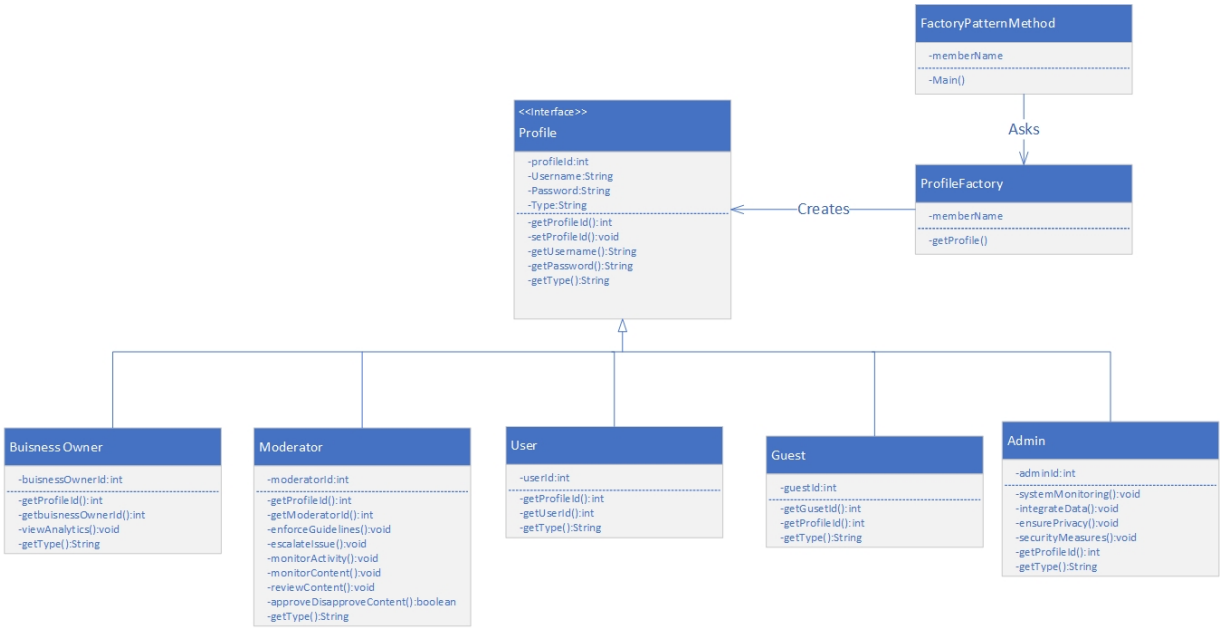
9. Design Pattern

UML design patterns describe reusable solutions to typical design challenges in software engineering. They offer an organized method to addressing design difficulties and encouraging best practices in software development. While design patterns are not UML diagrams in and of themselves, they may be used to demonstrate how they are implemented inside a system in conjunction with other UML diagrams such as class diagrams, sequence diagrams and component diagrams.

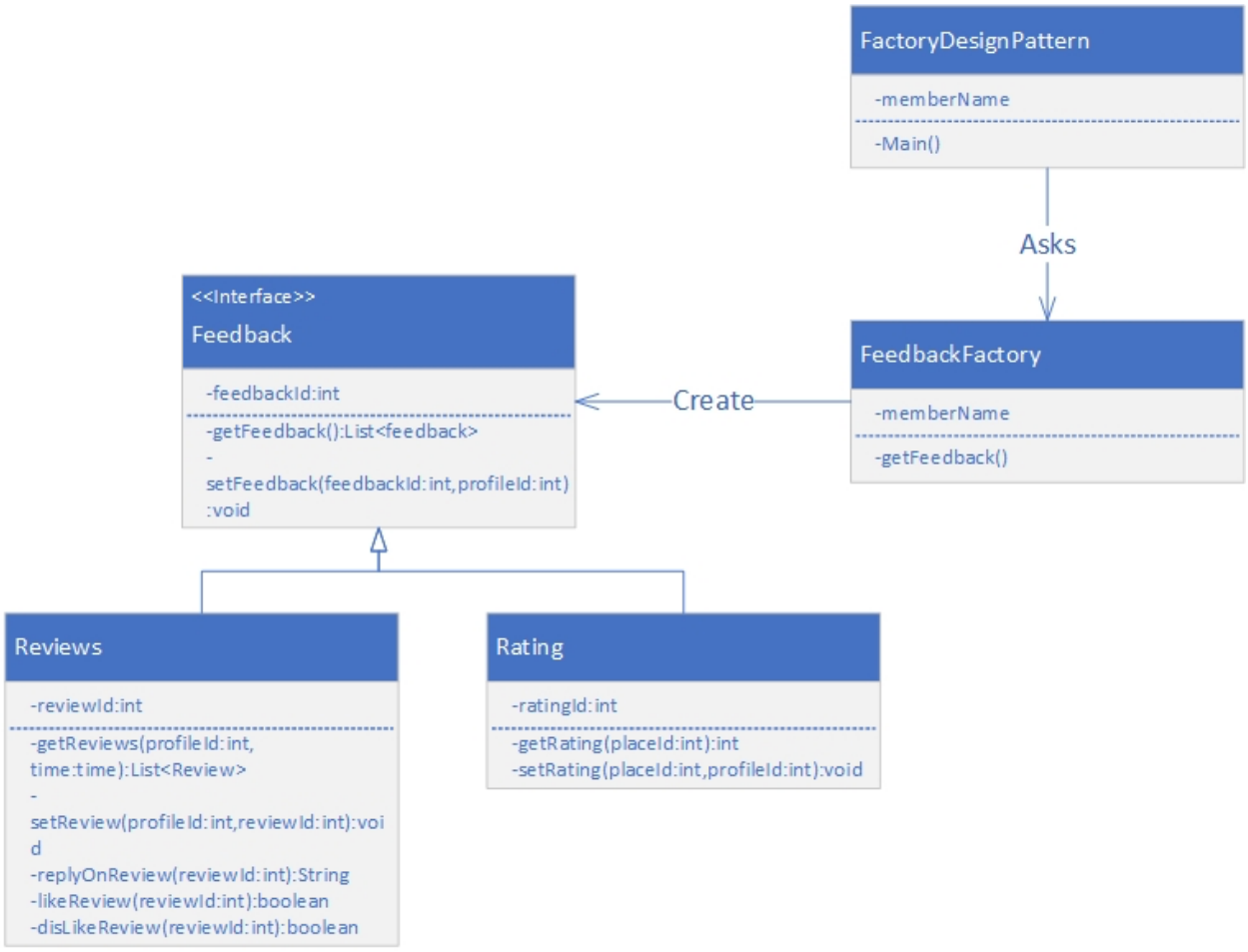
9.1 Factory Design Patterns

The Factory Design Pattern is a creational pattern that allows you to construct things without providing the specific class. Instead, it delegated responsibility for object creation to a separate factory class. This encourages loose connectivity between client code and produced objects, allowing for easier maintenance and greater flexibility in object generation. By enclosing the object generation logic within the factory class, the Factory Pattern allows for code scalability and the addition of new object types without altering current client code.

1. Profile Factory Design Pattern: The diagram depicts a Factory Design Pattern implementation for creating user profiles within a system. It displays an abstraction layer in which the Profile interface specifies common properties and functions, and a Profile Factory manages the creation of multiple user profiles such as Business Owner, Moderator, User, Guest, and Admin. This architecture isolates the creation logic, providing flexibility and scalability in user management while keeping the system decoupled and adhering to the idea of single responsibility. The structure guarantees that changes to user types have minimal impact on the existing system.



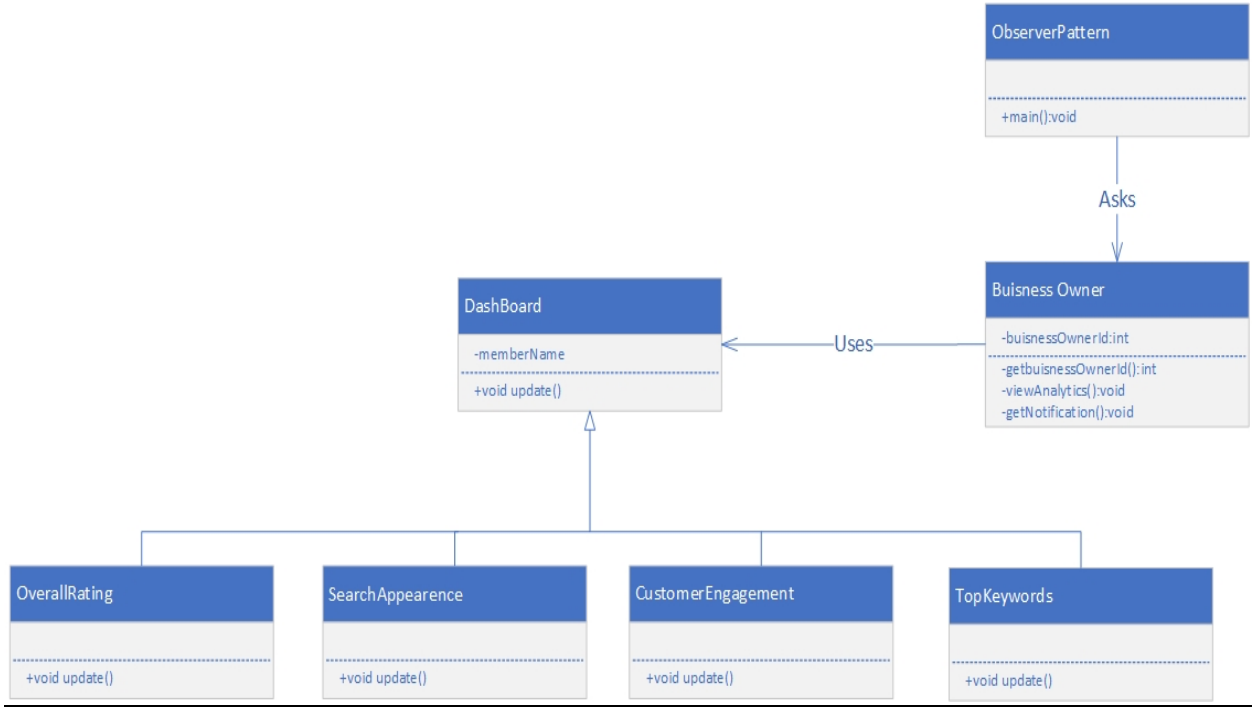
2. Feedback Factory Design Pattern: In this factory design pattern diagram, Feedback is represented as an interface with necessary methods for feedback manipulation, which are subsequently concretized by various sorts of feedback objects generated by the Feedback Factory. The Feedback Factory encapsulates the instantiation process and delegated the construction of individual feedback instances, guaranteeing that the system may respond to different forms of input while staying loosely connected. The associated Reviews and Rating classes provide a system that incorporates user feedback into larger quality measures, allowing users to access and change reviews and ratings, hence improving the system's interactive and evaluative capabilities.



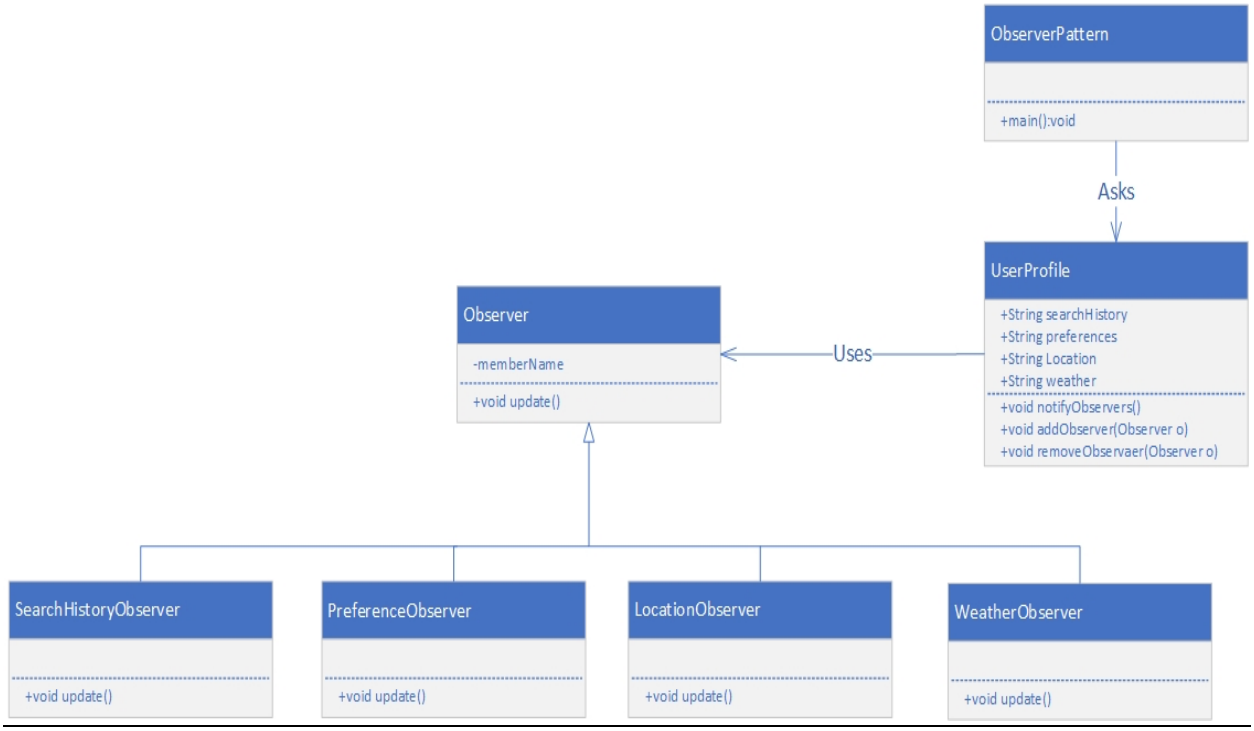
9.2 Observer Design Pattern

The Observer Design Pattern is a behavioral pattern that establishes a one-to-many dependency between objects, ensuring that when one object changes its state, all its dependents are notified and updated automatically. The subject (or observable) and the observers are the two main parts of this pattern. The observers register with the subject and receive updates corresponding to any changes in the subject's state, and the subject keeps track of all of them and alerts them of any changes.

1. Dashboard Observer Pattern: The image above depicts an Observer Design Pattern in which a Dashboard serves as the subject, with references to multiple observers such as Overall Rating, Search Appearance, Customer Engagement, and Top Keywords that monitor changes in state. The Business Owner may interact with the Dashboard to initiate modifications that are automatically reflected across all observers, guaranteeing real-time consistency across various KPIs. Each observer includes an update method, displaying a system that responds to changes and may alert key stakeholders about essential updates, resulting in a dynamic and interactive system design.



2. Preferences Observer Pattern: This observer pattern diagram depicts a user profile system that updates in real time as user data changes. The central Observer interface specifies a contract for updates that are carried out by observers for search history, preferences, location, and weather. These observers are used by a User Profile, which serves as the topic and notifies observers of changes. This approach guarantees that user-related changes are consistently and effectively propagated to appropriate subsystems, resulting in a highly responsive and synchronized user experience throughout the program.



9.3 Strategy Design Pattern

The Strategy Design Pattern is a behavioral pattern that allows for the dynamic selection of algorithms during runtime. It isolates each algorithm in distinct classes, allowing customers to select the right method without changing their code structure. By separating algorithms from client code and offering interchangeable strategies, the Strategy Pattern encourages code reuse, flexibility, and maintainability. This pattern is especially beneficial when there are numerous algorithms for accomplishing the same task or when the algorithm selection must change dynamically in response to changing requirements or situations.

1. Search Strategy Design Pattern: The image below depicts a Strategy Design Pattern for search functionality, with the abstract class Search Strategy providing a common interface for numerous search algorithms. Concrete classes that implement this interface include Search by Zip Code, Search by Rating, Search by Keyword, and Search By Price, all of which use specific algorithms for their particular search criteria. The Context class retains a reference to a Search Strategy, allowing it to swap strategies at runtime and execute the relevant search algorithm. This approach encapsulates search algorithms, offers interchangeable behaviors, and allows for the addition of search features without affecting the client code.